Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminar-Ausarbeitung Anwendungen 1

Stephanie Gamm

Transaktionen in verteilten und mobilen Systemen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Stephanie Gamm

Transaktionen in verteilten und mobilen Systemen

Seminar-Ausarbeitung im Rahmen der Veranstaltung Anwendungen 1
im Studiengang Master of Science Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Professor : Prof. Dr. Kai von Luck

Abgegeben am 19. Juli 2006

Stephanie Gamm

Thema der Seminar-Ausarbeitung Anwendungen 1

Transaktionen in verteilten und mobilen Systemen

Stichworte

verteilte Transaktionen, lang-laufende Transaktionen, mobile Transaktionen, Pervasive Gaming

Kurzzusammenfassung

Transaktionen stellen entscheidende Konzepte für verteilte Daten und verteilte Berechnungen zur Verfügung und sind heute ein wichtiger Bestandteil nahezu aller gängigen Anwendungssysteme. Durch die zunehmende Verbreitung tragbarer Rechner wie PDAs oder Laptops, die eine drahtlose Kommunikation unterstützen, gewinnen mobile Anwendungen zunehmend an Bedeutung. Mobile Systeme zeichnen sich jedoch durch spezielle Eigenschaften wie häufige Unterbrechungen der Netzwerkverbindung aus, wodurch sich die klassischen, bewährten Konzepte verteilter Transaktionen nicht einfach auf die mobile Umgebung übertragen lassen. Die vorliegende Arbeit beschäftigt sich mit ausgewählten, verteilten Transaktionsmodellen, um anschließend auf die besonderen Anforderungen an Transaktionen in mobilen Systemen aufmerksam zu machen. Dies geschieht mit Hinblick auf das im kommenden Semester geplante Projekt "Pervasive Gaming Framework", das ebenfalls kurz vorgestellt wird.

Inhaltsverzeichnis

1 Einleitung und Motivation

Transaktionen garantieren die Konsistenzerhaltung von Daten und stellen mittels verschiedener Konzepte ein Paradigma zur Behandlung von Fehlern zur Verfügung (GR-1993). Lange Zeit schienen die existierenden, ausgereiften Konzepte ausreichend, zumal heute kaum noch ein reales Anwendungssystem ohne die Unterstützung von Transaktionen auskommt.

Jedoch ist in jüngster Zeit wieder Bewegung in das Thema Transaktionen gekommen: Die zunehmend an Verbreitung gewinnenden, mobilen Geräte wie Personal Digital Assistents (kurz: PDAs) oder Laptops, die zumeist einen drahtlosen Zugang zu Netzwerken und somit mobiles Arbeiten ermöglichen, stellen neue Anforderungen an die Transaktionssysteme. Entgegen herkömmlichen, verteilten Transaktionen dürfen mobile Transaktionen insbesondere einen Verbindungsabbruch nicht als fatalen Fehler auffassen, sondern müssen diese Tatsache vielmehr als konzeptuelle Eigenart berücksichtigen.

Die vorliegende Arbeit will einige klassische Transaktionsmodelle und ihre Eigenschaften für verteilte Systeme aufzeigen, um darauf aufbauend die erweiterten Anforderungen mobiler Systeme, damit verbundene Probleme sowie erste konkrete Lösungsansätze für Modelle mobiler Transaktionen vorzustellen.

Das Kapitel 2 soll die notwendigen Grundlagen vermitteln. Während Kapitel 3 auf die Besonderheiten von Transaktionen bei verteilten Systemen eingeht, legt das Kapitel 4 den Fokus auf die mobilen Systeme. Abschließend gibt das Kapitel 5 einen Ausblick auf das geplante, weitere Vorgehen, besonders im Hinblick auf das studentische Projekt „Pervasive Gaming Framework“ im kommenden Semester, bei dem es sich um eine Ausprägung eines mobilen Systems handelt.

2 Grundlagen und Eigenschaften von Transaktionen

Eine Transaktion beschreibt eine Folge von Aktionen, die ausgeführt werden müssen, um ein Ziel zu erreichen. Dabei wird eine einzelne Transaktion stets als logische Einheit betrachtet.

Transaktionen haben ihren Ursprung im Bereich der Datenbanken. Anfang der 1970er begann die aktive Forschung auf diesem Gebiet. Man benötigte ein Kon-

zept, um bei Zugriff durch mehrere Clients die Konsistenz der serverseitigen Datenhaltung zu gewährleisten. In diesem Zusammenhang wurde 1983 durch Härder und Reuter der Begriff *ACID* geprägt, der die erwünschten Eigenschaften einer Transaktion wie folgt beschreibt (GR-1993):

Atomicity Eine Transaktion wird entweder ganz oder gar nicht ausgeführt.

Consistency Vor und nach jeder Transaktion befindet sich das Gesamtsystem in korrektem Zustand, d.h. Einhaltung sämtlicher Integritätsbedingungen.

Isolation Nebenläufig ausgeführte Transaktionen verhalten sich wie serielle Ausführung, d.h. keine gegenseitige Beeinflussung.

Durability Die Zustandsänderung einer erfolgreich beendeten Transaktion ist dauerhaft.

Im Laufe der Jahre wurde das klassische Konzept (vgl. dazu flache Transaktionen in Abschnitt 3.1) aufgrund erweiterter Anforderungen weiter entwickelt und vielfältige Transaktionsmodelle entstanden. Verteilte Transaktionen sind zu einer grundlegenden Technik geworden, die man in nahezu allen Anwendungssystemen findet. Dabei ist ihr Einsatz heute nicht mehr auf Datenbanken beschränkt; vielmehr stellen sie die entscheidenden Konzepte für verteilte Daten und verteilte Berechnungen zur Verfügung (Elm-1992). Aktuelle Transaktionsmodelle reichen von der Prozess-Kontrolle bis zu kooperativer Zusammenarbeit und sorgen dafür, dass sich das Gesamtsystem jederzeit in einem konsistenten Zustand befindet. Entgegen dem ursprünglichen Modell, wo viele Clients auf *einen* Server zugreifen, liegt der Fokus heutiger Modelle für verteilte Transaktionen auf der transparenten Verteilung auf *mehreren* Servern. Die wachsende Verbreitung mobiler Anwendungen macht es jedoch erforderlich, dass neuartige Konzepte entwickelt und für den Einsatz mit mobilen Geräten praxistauglich gemacht werden.

3 Transaktionsmodelle für verteilte Systeme

In der Literatur lassen sich zahlreiche Transaktionsmodelle finden, die den verschiedenen Situationen und Anforderungen verteilter Systeme angepasst sind. Die folgenden zwei Abschnitte sollen einen Einblick in diese Modelle geben, dabei dient Abschnitt 3.1 als Einstieg in die klassischen, verteilten Transaktionen und zeigt grundlegende Konzepte auf. Abschnitt 3.2 beschäftigt sich speziell mit langlaufenden Transaktionen.

3.1 Klassische, verteilte Transaktionen

Bei der einfachsten verteilten Transaktion (engl. *distributed transaction*) handelt es sich um eine flache Transaktion (engl. *flat transaction*) in verteilter Umgebung. Wesentlich für diesen Transaktionstyp ist, dass ein Knoten den gesamten internen Ablauf kontrolliert. Die einzelnen Aktivitäten werden sequentiell ausgeführt, wobei dies auf verschiedenen Knoten im Netzwerk geschieht. Die flache verteilte Transaktion erfüllt alle ACID-Eigenschaften und gehorcht somit dem „Ganz-oder-gar-nicht“-Prinzip. Typisch für flache Transaktionen ist, dass sie schnell abgeschlossen sind; ihre Ausführung ist meist nur von sehr kurzer Dauer. Für neuere Arten von Anwendungen stellte dies jedoch eine Schwachstelle dar, so dass man spezialisierte Transaktionsmodelle entwickelte.

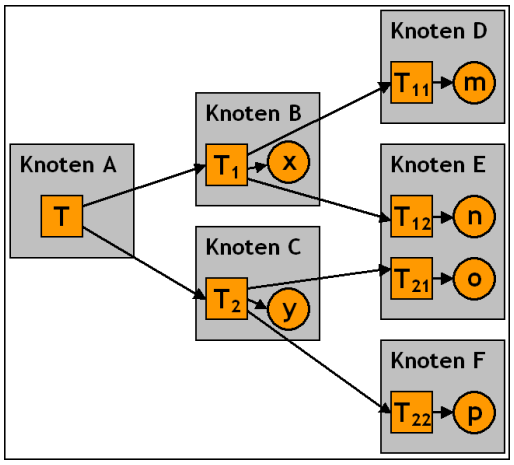

Abbildung 1: Nested Transaction

Eine Abwandlung sind die *Nested Transactions* (GR-1993), (WV-2002). Hierbei handelt es sich um ineinander geschachtelte Subtransaktionen, die intern eine hierarchische Anordnung – analog einer Baumstruktur – erlauben. Abbildung 1 veranschaulicht beispielhaft den Aufbau einer Nested Transaction.

Um der Ineffizienz, wie sie bei flachen Transaktionen aufgrund der internen, sequentiellen Ausführung auftritt, entgegen zu wirken, können bei der Nested Transaction alle Kind-Transaktionen einer Eltern-Transaktion sowohl sequentiell wie auch parallel ausgeführt werden. Das Commit aller Subtransaktionen geschieht weiterhin erst bei Commit der Root-Transaktion. Aufgrund des geschachtelten Aufbaus besteht die Möglichkeit der Modularisierung bzw. Einführung von Abstraktionsebenen innerhalb einer Transaktion.

Ein anderes Transaktionsmodell sind die *Chained Transactions* (GR-1993), (WV-2002). Diese erlauben das Commit von Zwischenergebnissen, um erfolgreich absolvierte Aktivitäten bereits vor Ablauf der gesamten Transaktion zu sichern. Ihr Aufbau entspricht einer sequentiellen Verkettung von Subtransaktionen. Chained Transactions realisieren das sogenannte *Chaining*: Das Commit einer Subtransaktion startet automatisch die nächste Subtransaktion.

Durch frühes Freigeben der gesperrten Ressourcen wird bei Chained Transactions eine bessere Performance erreicht. Im Fehlerfall wird das Rollback nur für die aktuelle Subtransaktion ausgeführt. Dies führt jedoch zu dem Problem, dass der Initiator der gesamten Transaktion im Falle eines Fehlers keine Kenntnis über die erreichte Zustandsänderung erhält, da die abgebrochene Transaktion diese Information nicht nach außen weiter reicht. Die erforderlichen Recovery-Maßnahmen zum erfolgreichen Beenden oder Fortführen der unterbrochenen Transaktion können nicht ausgeführt werden, womit das gesamte System wieder in einen konsistenten Zustand überführt würde.

3.2 Lang-laufende, verteilte Transaktionen

Eine weitere Klasse innerhalb der existierenden Transaktionsmodelle bilden die lang-laufenden Transaktionen (engl. *long-lived* oder auch *long-running transactions*). Sie genügen entgegen den in Abschnitt 3.1 vorgestellten klassischen Transaktionen erweiterten Anforderungen, wie verstärkter Verteilung und Integration. Ihr Einsatzgebiet erstreckt sich besonders auf heterogene Systemlandschaften.

Lang-laufende Transaktionen sind für lange Laufzeiten optimiert, so dass beispielsweise eine Interaktion von Mensch und Maschine innerhalb einer globalen Transaktion mit einer Dauer von Stunden, Tagen oder sogar Monaten zufrieden stellend abgearbeitet werden kann. Für derartige Szenarien sind die klassischen Transaktionen ungeeignet, da aufgrund der Locking-Mechanismen wie dem 2-Phase-Locking die Ressourcen zu lange gesperrt werden.

Um im Fehlerfall möglichst wenig erfolgreiche Arbeit zu verlieren, erlauben langlaufende Transaktionen das Commit von Zwischenergebnissen. Ihr interner Aufbau besteht aus einer Komposition mehrerer, lose gekoppelter Subtransaktionen, so dass als Basis oftmals die in Abschnitt 3.1 vorgestellten Chained Transactions verwendet werden. Das Commit von Zwischenergebnissen führt dazu, dass interne Zwischenzustände schon vor Ablauf der Transaktion für andere Teilnehmer sichtbar werden. Dieses Lösen von der geforderten Eigenschaft Atomicity geht

man jedoch bewusst ein, um den erweiterten Anforderungen gerecht werden zu können. Typisch für lang-laufende Transaktionen ist somit das bewusste Lockern der ACID-Eigenschaften (vgl. dazu Kapitel 2) in verschiedenen Ausprägungen sowie eine optimistische Nebenläufigkeitskontrolle.

Damit Transaktionen in realen Anwendungssystemen einsetzbar sind, dürfen sie nicht dasselbe Problem wie die Chained Transactions bei Unterbrechung einer Transaktion aufweisen (vgl. Abschnitt 3.1). Bei lang-laufenden Transaktionen besteht daher die Möglichkeit, sich den Aufsetzpunkt im Fehlerfall außerhalb der Transaktion zu merken. Nach (GR-1993) existieren für das anschließende Recovery zwei mögliche Strategien: das Fortsetzen der Transaktion am Aufsetzpunkt oder das Rückgängig machen aller bisherigen Aktionen.

Ein Modell für das Fortsetzen der unterbrochenen Transaktion ist der so genannte *Mini-Batch* (GR-1993). Grundlage für den Mini-Batch bildet eine Chained Transaction, wobei ein explizites Merken der erfolgreich ausgeführten Aktionen hinzukommt. Im Fehlerfall kann wieder in den Ablaufplan eingegriffen und die Transaktion am Aufsetzpunkt fortgesetzt werden. Ein Rollback wird höchstens für die unterbrochene Subtransaktion, nicht aber für die gesamte Transaktion ausgeführt.

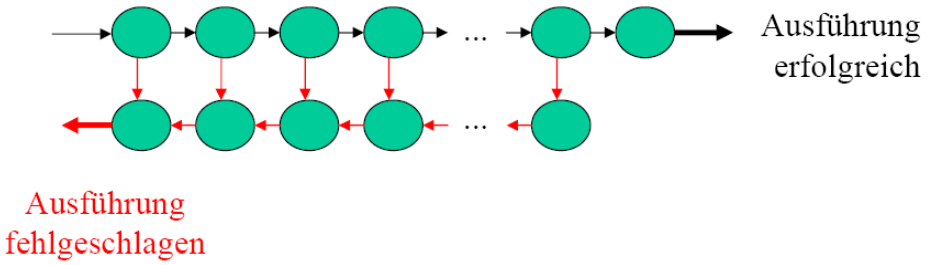

Abbildung 2: Kompensation bei Sagas (Vos-2006)

Die zweite Recovery-Strategie setzen die *Sagas* um (GR-1993), (KBL-2006). Auch ihnen liegt das Modell der Chained Transactions zugrunde. Zusätzlich wird für jede Subtransaktion die Definition einer entsprechenden *Kompensations*-Transaktion (engl. *Compensation*) benötigt, um im Fehlerfall alle bisherigen Aktionen rückgängig machen zu können. Mit *Kompensation* wird die semantisch inverse Aktion zu einer Aktion bezeichnet; zu der Aktion *Buchbestellung* wäre beispielsweise *Bestellstornierung* die Kompensations-Aktion. Bei Abbruch einer Subtransaktion werden die Kompensations-Aktionen aller vorherigen Subtransaktionen in umgekehrter Reihenfolge ausgeführt. In diesem Fall wird für die gesamte Transaktion ein Rollback ausgeführt. Abbildung 2 zeigt die möglichen Abläufe für den erfolgreichen wie auch fehlgeschlagenen Fall auf.

Neben Mini-Batch und Sagas existieren in diesem Kontext weitaus mehr Modelle für lang-laufende, verteilte Transaktionen. Eine besonders aktuelle Entwicklung sind *Workflows*, wobei es sich um die Implementierung komplexer, lang-laufender Geschäftsprozesse in verteilter und heterogener Umgebung handelt (KBL-2006), in Verbindung mit der *Service Oriented Architecture* (kurz: SOA). Hier liegt der Fokus auf der Implementierung von Web Services und deren Orchestrierung. Eine strenge Anforderung an die ACID-Eigenschaften besteht für gewöhnlich nicht.

Diese Arbeit kann jedoch nur einen Auszug der wichtigsten Konzepte exemplarisch aufzeigen, daher wird für weitergehende Informationen auf die einschlägige Fachliteratur verwiesen.

4 Transaktionsmodelle für mobile Systeme

Existierende Transaktionskonzepte für verteilte Systeme (vgl. dazu Kapitel 3, (GR-1993) und (WV-2002)) genügen den Anforderungen mobiler Systeme nicht. Abschnitt 4.1 soll die spezifischen Eigenschaften und die damit verbundenen Anforderungen mobiler Systeme vermitteln, während Abschnitt 4.2 einen Einstieg in darauf spezialisierte Transaktionsmodelle gibt.

4.1 Eigenschaften und Anforderungen mobiler Systeme

Die zunehmende Verbreitung tragbarer Rechner wie PDAs, Laptops oder Mobiltelefonen in Verbindung mit der steigenden Verfügbarkeit drahtloser Netzwerke lässt auch den Einsatz mobiler Anwendungssysteme an Bedeutung gewinnen. Dabei möchten die Nutzer dieser Systeme sich nicht auf die isolierten, also rein lokal vorhandenen Daten beschränken, sondern beispielsweise unbeschränkt auf Informationen bestehender Anwendungssysteme des Firmennetzes zugreifen können. Prinzipiell sollte es möglich sein, Informationen zu jeder Zeit und an jedem Ort abzurufen.

In Kapitel 3 wurden Transaktionsmodelle für verteilte Systeme betrachtet; dort steht die transparente Verteilung von Daten oder Berechnungen im Vordergrund. Verteilte Transaktionen gehen von einer dauerhaften Konnektivität, insbesondere von einer feststehenden Netz-Topologie mit entsprechender Verfügbarkeit und Zuverlässigkeit der beteiligten Systeme aus. Eine unterbrochene Netzwerkverbindung bedeutet eine Fehlersituation und wird entsprechend behandelt.

Durch den Einsatz mobiler Geräte und die damit verbundene Dynamik lassen sich die bewährten Konzepte klassischer, verteilter Systeme nicht einfach auf mobile Systeme übertragen. Neben allgemeinen Anforderungen an mobile Anwendungen, die aufgrund der oftmals eingeschränkten Leistungsfähigkeit mobiler Geräte bestehen, stellt die *Mobilität* der teilnehmenden Systeme einen wesentlichen Aspekt dar, den es zu berücksichtigen gilt. Knoten des Gesamtsystems können nun auch mobile Knoten und dadurch nicht ständig mit den anderen Teilnehmern fest verbunden sein. Transaktionsmodelle, die sich auf diese Anforderungen spezialisiert haben, werden unter dem Begriff *mobile Transaktionen* zusammengefasst und sollen im folgenden näher vorgestellt werden.

Aufgrund der Rahmenbedingungen, denen mobile Transaktionen unterliegen, ist ein striktes Einhalten der ACID-Anforderungen nicht möglich, da viel zu häufig Rollbacks ausgeführt werden müssten. Stattdessen müssen die Konzepte auf die Bewegung der Knoten sowie den Abbruch der Verbindung ausgelegt sein.

Ist der mobile Client bis auf kurze Ausnahmen mit dem Netzwerk verbunden, spricht man von schwach verbundenen Systemen. Als Beispiel hierfür wäre ein Mitarbeiter denkbar, der mit seinem PDA durch das Firmengebäude läuft und dabei über wechselnde WLAN-Basisstationen mit dem Firmennetzwerk verbunden ist. Der PDA ist hier nahezu ständig online – mit Ausnahme der Wechsel in eine andere Funkzelle. Demgegenüber steht ein Szenario, bei dem der Aspekt Mobilität noch deutlicher wird. In den meisten Fällen wird es sich vermutlich nur um temporäre Netzwerkverbindungen handeln. Der mobile Client kann hierbei für einen längeren Zeitraum – von möglicherweise Stunden oder Tagen – nicht mit dem übrigen System verbunden sein und unterliegt einem ständig wechselnden Online/Offline-Szenario. Dies wäre beispielsweise der Fall, wenn der Mitarbeiter den PDA auch auf Kundenbesuche oder nach Hause mitnimmt und erst wieder in der Firma eine entsprechende WLAN-Verbindung aufbauen kann.

Eine wesentliche Eigenschaft mobiler Systeme besteht demnach darin, dass die Verfügbarkeit und Zuverlässigkeit der beteiligten Systeme stark eingeschränkt ist (Ger-2006). Der mögliche, meist nicht vorhersehbare Abbruch bzw. die Unterbrechung der Netzwerkverbindung während einer Transaktion muss bei der Entwicklung geeigneter Konzepte berücksichtigt werden. Um entsprechend reagieren zu können, muss zunächst die Unterbrechung festgestellt werden, wobei meist ungewiss ist, wie lange die Unterbrechung anhalten wird bzw. ob es sich sogar um einen Totalausfall handelt.

Weiterhin muss bedacht werden, dass ein für eine Subtransaktion zuständiger Knoten möglicherweise aktuell nicht erreichbar ist. In diesem Zusammenhang

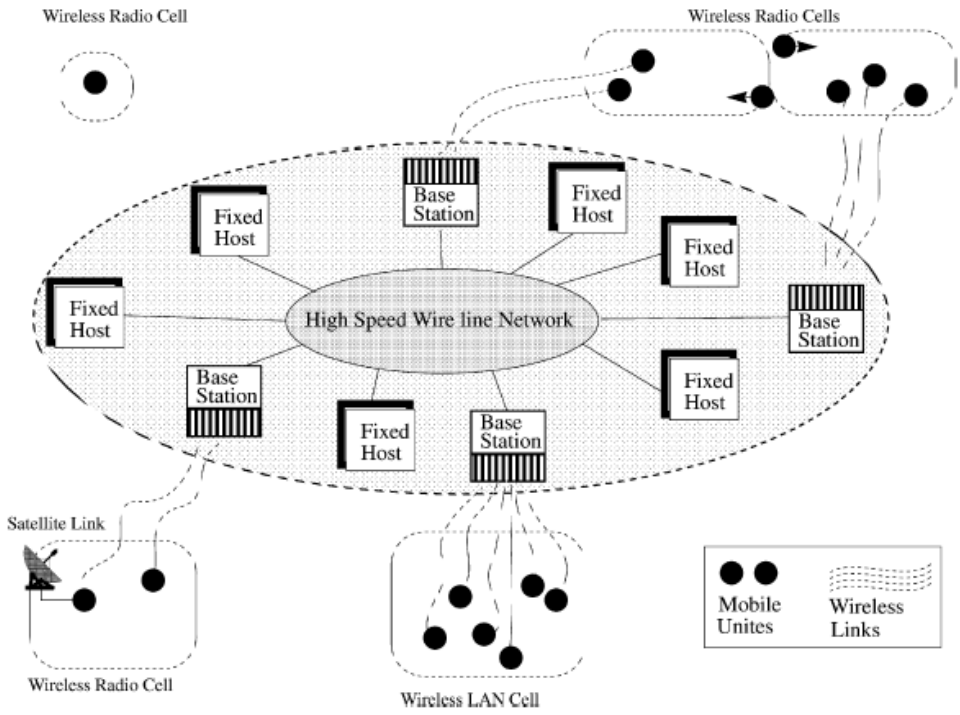

Abbildung 3: Mobiles System (DHB-1997)

muss geklärt werden, ob die globale Transaktion ggf. auch ohne dessen Zustimmung erfolgreich abgeschlossen werden kann und ob dieser Knoten über die angeforderte Subtransaktion dennoch informiert wird, sobald er wieder online ist. Denkbar wäre auch eine Kompensationsstrategie. Jedoch ist dann zu entscheiden, was mit Knoten geschieht, die zuvor dem Commit zugestimmt haben und nun für die Kompensation nicht mehr zu erreichen sind. Dabei sollten derartige Entscheidungen möglichst automatisch getroffen werden können und nur, wenn unbedingt notwendig, den Eingriff des Anwenders erfordern.

Grundsätzlich müssen mobile Transaktionen nach (KR-2002) folgende Eigenschaften unterstützen: den nahtlosen Übergang von einer Zelle zur anderen, die Wiederaufnahme von Arbeiten oder Rücksetzen nach versehentlichem Verbindungsabbruch, die Wiederaufnahme von Arbeiten nach absichtlichem Verbindungsabbruch, die Verteilung von Arbeitsschritten zwischen mobilem Gerät und Basisstation(en) sowie langlaufende Transaktionen.

Versteht man mobile Geräte als Erweiterung eines klassischen, verteilten Systems, so wird man die mobilen Systeme als zusätzliche Clients mit einem wei-

terhin stationären Server (oder Servern) betrachten. Eine weitere Möglichkeit ergibt sich, indem die mobilen Systeme selbst Services zur Verfügung stellen (vergl. dazu (Ger-2006)) und somit im Rahmen einer SOA als mobile Server dienen. In diesem Kontext spielt ebenfalls das Auftauchen und Verschwinden von Services eine Rolle. Allgemein gilt es für die Entwicklung mobiler Transaktionen, viele Eigenschaften eines Ad-hoc-Netzwerkes zu berücksichtigen.

Um mobiles Arbeiten trotz längerer Offline-Zeiten zu ermöglichen und den Datenzugriff auch ohne aktive Netzwerkverbindung gewährleisten zu können, wird man oftmals auf redundante Datenhaltung zurückgreifen müssen. Das Vorhalten redundanter Daten auf verschiedenen Knoten bezeichnet man als *Replikation*. Die Einführung von Replikation führt dazu, dass die Konsistenzerhaltung der replizierten Daten erschwert wird. Exemplarisch sei hier das Zusammenführen lokal ausgeführter Transaktionen auf dedizierten Master-Daten erwähnt. Das Gebiet, das sich mit diesem Thema beschäftigt, ist die *Synchronisation*. Sowohl Replikation wie auch Synchronisation spielen oftmals eine wesentliche Rolle bei mobilen Anwendungen und somit auch bei vielen mobilen Transaktionen.

4.2 Mobile Transaktionen

Existierende Modelle für mobile Transaktionen lassen sich in zwei Kategorien einteilen (Tho-2005), die danach unterscheiden, ob der mobile Client schwach mit dem Gesamtsystem verbunden ist oder ob zum Zeitpunkt der Transaktion ggf. gar keine Verbindung besteht und die Änderungen später automatisch nachgezogen werden müssen.

Ein Transaktionsmodell für die erste Kategorie sind die *Kangaroo-Transaktionen*. Sie berücksichtigen die Bewegung des mobilen Clients während der Dauer einer Transaktion, z.B. beim Wechsel einer Funkzelle. Dabei wird die gesamte Transaktion in Subtransaktionen aufgeteilt und deren Ausführung auf die jeweils beteiligten Basisstationen verteilt. Schlägt eine Subtransaktion fehl, so existieren bei Kangaroo-Transaktionen zwei Möglichkeiten (KR-2002): Entweder wurden für alle Aktionen entsprechende Compensation-Transaktionen definiert und diese werden für alle bereits erfolgreich abgeschlossenen Subtransaktionen ausgeführt, womit die gesamte Transaktion rückgängig gemacht wird. Andernfalls kann ein Wiederanlauf durch Fortsetzen der begonnenen Transaktion gestartet werden.

Modelle der zweiten Kategorie sind am besten für das in Abschnitt 4.1 beschriebene Online/Offline-Szenario einzusetzen. Bei der *Provisorischen Transaktion*, die

dieser Kategorie zuzuordnen ist, wird zwischen mobilen und stationären Knoten unterschieden. Ist ein mobiler Knoten während der Transaktion nicht mit dem Netz verbunden, d.h. offline, werden die Änderungen provisorisch auf dem lokalen, replizierten Datenbestand ausgeführt. Ist der mobile Knoten zu einem späteren Zeitpunkt wieder online, wird diese Transaktion im Rahmen der Synchronisation auf dem stationären Knoten nachgezogen. Dazu existiert ein ausgewiesener Master-Knoten, der die Änderungen anschließend an alle anderen Knoten propagiert.

5 Fazit und Ausblick

Die Mehrheit der in der Theorie existierenden Transaktionskonzepte für verteilte Systeme sind ausgereift und in kommerziellen Produkten implementiert bzw. als Standard anerkannt. Dennoch kommen immer wieder neue Anwendungsbereiche hinzu, wo noch Handlungsbedarf besteht, um die theoretisch erarbeiteten Modelle praxistauglich zu machen – wie es beispielsweise speziell bei hoher Verteilung und heterogenen, lose gekoppelten Systemen wie der SOA der Fall ist – oder es müssen sogar ganz neue Konzepte erarbeitet werden, wie es aktuell bei den mobilen Systemen gegeben ist.

Eine Art der mobilen Systeme mit besonders neuartigen Eigenschaften in Bezug auf die Entwicklung von Anwendungssystemen stellt der Bereich *Pervasive Gaming* dar (BML-2005). Auf diesem Hintergrund soll im Rahmen des Projektes „Pervasive Gaming Framework“ im kommenden Semester ein Framework für ein mobiles, mehrbenutzerfähiges Spiel entworfen und prototypisch realisiert werden.

Die Autorin plant, an der Entwicklung einer geeigneten Framework-Architektur für Pervasive Gaming mitzuwirken mit besonderem Hinblick auf die Integration transaktionsorientierter Komponenten. Es sollen unter Berücksichtigung der in Abschnitt 4.1 erläuterten Eigenschaften mobiler Systeme für das Umfeld mobiler Spiele geeignete Transaktionsmodelle evaluiert und weiterhin untersucht werden, inwieweit aufgrund zu erwartender, längerer Offline-Zeiten Replikation für einen ungestörten Spielfluss eingesetzt werden muss. Ggf. müssen transaktionale Operationen zu stationären und mobilen Servern, wie auch untereinander, zwischen sich treffenden Clients unterstützt werden.

Weitere Themen, die im Rahmen dieses Projektes von Kommilitonen untersucht werden sollen, sind Location Based Services, Sicherheit, kollaborative Bearbeitung von Dokumenten sowie Autonomic Computing.

Literatur

[BML-2005] BENFORD, Steve ; MAGERKURTH, Carsten ; LJUNGSTRAND, Peter: Bridging the physical and digital in Pervasive Gaming. In: *Communications of the ACM, Vol. 48, No. 3* (2005)

[DHB-1997] DUNHAM, Margaret H. ; HELAL, Abdelsalam ; BALAKRISHNAN, Santosh: *A mobile transaction model that captures both the data and movement behavior*. Version: 1997. http://nenya.ms.mff.cuni.cz/related/publ/tp/p149-dunham.pdf, Abruf: 16.07.2006

[Elm-1992] ELMAGARMID, Ahmed K. (Hrsg.): *Database transaction models for advanced applications*. San Mateo, California : Morgan Kaufmann Publishers, 1992

[Ger-2006] GERLACH, Martin: *Entwicklung eines Transaktions-Frameworks für mobile Web Services*. HAW Hamburg, Masterarbeit (noch in Arbeit), 2006

[GR-1993] GREY, Jim ; REUTER, Andreas: *Transaction processing: concepts and techniques*. San Francisco, California : Morgan Kaufmann Publishers, 1993

[KBL-2006] KIFER, Michael ; BERNSTEIN, Arthur ; LEWIS, Philip M.: *Database systems – An application-oriented approach*. Second edition, Complete version. Boston, San Francisco, New York [...] : Addison Wesley, 2006

[KR-2002] KÖNIG-RIES, Birgitta: *Mobile Transaktionen*. Version: 2002. http://www.ipd.uni-karlsruhe.de/~koenig/MODIS/transaktionen.pdf, Abruf: 16.07.2006

[Tho-2005] THOMÉ, Mark: *Mobile Datenbanksysteme*. Version: 2005. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/thome/abstract.pdf, Abruf: 16.07.2006

[Vos-2006] VOSSEN, Gottfried: *Workflows und transaktionale Garantien*. Version: 2006. http://www.wi.uni-muenster.de/imperia/md/content/wi-information_systems/lehrveranstaltungen/lehrveranstaltungen/bpmundwfm/ws0506/2006_01_09.transaktionale.garantien.pdf, Abruf: 13.07.2006

[WV-2002] WEIKUM, Gerhard ; VOSSEN, Gottfried: *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*. San Francisco, California : Morgan Kaufmann Publishers, 2002